Value Stream Maps and Management

This paper is derived from selections from my book.



The book “Engineering DevOps” can be obtained here: [**mybook.to/engineeringdevops**](http://mybook.to/engineeringdevops)

This white paper and other useful papers, glossary checklists, slides, blueprints, templates and engineering practices can be downloaded from [**https://engineeringdevops.com/documents**](https://engineeringdevops.com/documents)

About the Author:

**Marc Hornbeek** a.k.a. DevOps\_the\_Gray esq.

[linkedin.com/in/marchornbeek](https://www.linkedin.com/in/marchornbeek)

May 8, 2020

© Engineering DevOps Consulting 2019, 2020

All rights reserved

**How do you engineer value stream maps and manage value streams?**

In this whitepaper, we will look at a lean value stream pipeline engineering techniques approach to breaking down complex projects into executable project components. And we will also look at a blueprint, tools and best practices for managing value streams.

**Lean DevOps Value-Stream Pipeline Engineering**

Modern supermarkets and shopping malls are amazing places. The giant array of products for affordable prices, all in one place, makes the average consumer take for granted their abundance and convenience. But how did all these products get there? How did they come into existence at all? In the movie Six Days and Seven Nights,RW19 co-stars Anne Heche and Harrison Ford crash land on a deserted island. While lamenting the lack of modern conveniences, Anne says to Harrison, “Aren’t you one of those guys? You know those guys with skills. Yeah, you send them out into the wilderness with a pocket knife and a Q-tip and they build you a shopping mall. You can’t do that?”

I love that quote. It makes you stop and think about cool stuff like invention, design, supply chains, production, distribution networks, and the incredible number of engineering steps involved to move a product from idea to the cash register—a long chain in which each step adds value. The DevOps continuous delivery pipeline implements a value stream for software. ***Figure 1—DevOps Value-Stream Pipeline Example*** illustrates this idea. Not all value streams and pipelines are the same. There is no standard. A tool for visualizing value streams in an easy-to-understand series of process steps has been used in lean manufacturing and business process engineering for years. It is called Value-Stream Mapping. Value-Stream Mapping helps identify things important to understanding how DevOps works.



**Figure 1: DevOps Value Stream Pipeline Example**

Wouldn’t it be great if you could just think about a product and voila, the perfect product appears on the store shelf in front of you? You are a dreamer, too! Unfortunately, in the real world it takes time.

***Figure 2— Value-Stream Map Template*** illustrates some of the key ideas for understanding how value-stream maps are used to identify bottlenecks in the value stream for DevOps pipelines. Bottlenecks come in various flavors, and none of them taste good. Lead time and quality of process steps and series of process steps are two of the most interesting bottlenecks that value stream engineering is concerned with. Lead Time (LT) is the duration that a process requires from entry to exit. Process Time (PT) is the time it takes to perform the work of the process. PT is usually less than LT because processes are rarely 100% efficient. Within a process, there are often things happening that are considered “wasted time” from a process point of view. An example could be a lunch break. In DevOps, waste could be internal wait times for resources to be orchestrated before a build or test can be performed. There can also be waste between process steps. This is referred to Non-Value-Time (NVT) in the diagram. Examples of NVT for DevOps value stream is the time waiting for a meeting to get approvals to allow software changes to be promoted to the next stage in the pipeline. Did we forget about quality already? Shame on you. Fortunately, values stream maps never forget. %C/A is a ratio, expressed as a percentage. It is used to measure processes that use the results of the prior process without requiring rework. What do C and A stand for? I don’t remember, but it’s not important. What is important is that rework is a form of waste and a bottleneck that tastes bad to consumers.



**Figure 2: Value Stream Map Template**

Let us look at a real example of how this works. ***Figure 3—Value- Stream Map Example*** is an actual value-stream map that I worked on together with a team that is responsible for a policy management software application of a large insurance company. Each stage of the value-stream map identifies the process stages, the content of each stage, and the LT, PT, NVT, and %C/A values. It sounds easy when I say that in one sentence like that. Getting to this simple map takes a lot of work! A glance at the value-stream map shows where there are key bottlenecks. Or does it? The length of time for each process is only part of the story. Waste is something else. Let us think about this. A simple calculation (LT-PT)/LT for each stage is the percent efficiency for each stage. By adding all the LT and PT and NVT values, you can calculate end-to-end percent efficiency. The numbers highlighted in ***Figure 4—Value-Stream Map Lead Time Example*** are significant to lead time improvement because they point out processes and inter-process gaps that have a high level of waste and a low level of inefficiency. These are areas to target solutions to reduce waste and improve overall lead time.

The numbers and processes highlighted in the ***Figure 5—Value- Stream Map—Quality Example*** are important to quality improvement. Remember that %C/A, whatever it stands for, is a measure of the percentage of time that a process stage can use results of prior stage without rework. The series of %C/A values can be multiplied to get an end-to-end %C/A value. In this example, only 15% of the changes committed at the entry gate to the Dev process stage make it all the way to the Operate process stage without requiring rework. What a waste! What can be done? Looking at which process stages are causing the most rework is a good place to look for solutions.



**Figure 3: Value Stream Map Example**



**Figure 4: Value Stream Map Lead Time Example**



**Figure 5: Value Stream Map Quality Example**

So how is this used for DevOps? Let’s say you have a specific DevOps goal to reduce your overall lead time from 255 hours to 200 hours per release (55 hours less than the current state) while improving quality by increasing the percentage of changes that do not require rework from 15% to 25% (10 points of improvement from current state). The Value-Stream Analysis tells you which stages and stage transitions to target for process and quality bottlenecks that are going to yield the largest improvements. The current state value stream analysis shown in ***Figure 4—Value-Stream Map Lead Time*** Example and ***Figure 5—Value-Stream Map—Quality Example*** indicate that improvements to CI and Delivery stages and stage hand-offs will most easily yield lead time improvements, while improvements to the Dev, CI, and Deployment stages will most easily yield quality improvements. The value stream current state analysis does not, by itself, tell you specific improvements to make, but it guides you to the areas of the pipeline to best concentrate engineering solutions that will move towards improvement the quickest.

After the current state DevOps value stream is in place, the next step is to create a future state DevOps value-stream map that will describe changes to the pipeline so that it will perform according to the organizations’ s DevOps goals. Creating a future state DevOps value-stream map requires DevOps engineering knowledge and expertise to design specific solutions.

An example future state value-stream map is shown in the ***Figure 6—Value-Stream Map—Future State Example***.

To accomplish the goal of reducing the end-to-end lead time by at least 50 hours, the future state DevOps value-stream map indicates 10 hours lead time reduction to the Dev, CI, and Deliver stages and 6 to 14 hours reduction to most of the stage wait times.

To accomplish the goal of improving quality by at least 10 points, the future state DevOps value-stream map indicates %C/A increases of 10 points for CI and Deliver stages and 20 points for the Operate stage, resulting in and end-to-end %C/A of 26% compared to the current state of 15%. The future state DevOps Value-Stream Map indicates the lead time reduction and quality improvements can be accomplished by implementing the following changes:

• Backlog stage: Include test plan automation tasks in backlog

• Dev stage: Automate unit testing and use Test-Driven Development for functional test creation

• CI stage: Automate CI builds and smoke and regression tests

• Deliver stage: Automate release acceptance tests

• Deploy stage: Automate CAB approvals using ARA

• Between stages: Orchestrate build and test environments; use containers

This list of changes represents a solution from a DevOps expert knowledgeable and experienced in DevOps solutions and recommended engineering practices. There are usually multiple possible solutions to improve DevOps improvements. A DevOps expert considers multiple factors when selecting solutions for a specific value stream and goal. These are discussed in more depth in other white papers in this series in the book ***Engineering DevOps***. DevOps solutions are implemented incrementally in accordance with DevOps tenets and practices! Use DevOps for DevOps! After each increment, results are measured to validate the solution is moving towards the goal. Incremental changes continue until the goal is reached. Once the DevOps goals are accomplished, new DevOps goals are set and a new current state/future state value stream analysis is conducted.



**Figure 6: Value Stream Map Future State Example**

Lean value-stream pipeline engineering sits at the top of the DevOps Engineering Blueprint because this is the layer that governs the entire end-to-end value stream from conception through operations. In the next chapter I explain engineering practices and technologies that assist in managing the end-to-end value stream.

**Value-Stream Management (VSM)**

VSM solutions implement an abstraction layer over one or more end-to-end value streams from planning through to operation and all the stages in the middle. VSM is especially powerful when an organization has multiple value streams or pipelines to manage.

**Why Is Value-Stream Management Important to DevOps?**

VSM solutions address the following problems that are prevalent in “Second Way” and “Third Way” DevOps implementations:

• Fragmented visibility along each pipeline from planning through operations

• Lack of visibility of dependencies across product portfolios and multiple pipelines

• Work-in-progress and work-in-aggregate tracking

• Optimization across disparate pipelines

• Facilitate methodologies for continuous improvement

How Does Value-Stream Management Work with DevOps?

As shown in ***Figure 7—Value Stream Management Blueprint***, VSM manages the application delivery process from inception to delivery across the entire enterprise portfolio of multiple value streams. VSM supports DevOps governance, notification, insights, and analytics, as well as orchestration across teams, in a way that would be difficult to accomplish without VSM tools.



**Figure 7: Value Stream Management Blueprint**

As shown in ***Figure 8—Value-Stream Management Tool***, VSM tools are key to supporting support release planning, coordination and orchestration, and shared environments management taking a broader view of releases to include from planning all the way through deployment and connecting the points in between. It is possible to build these capabilities from ARA or DevOps toolchains, but VSM tools have the advantage of being designed for this right out of the box. VSM tools offer governance and value regardless of the mix of ARA tools or even development methodology, making them an enabling technology for managing the digital transformation journey. Value-stream management solutions provide the following three key areas of functionality:

1. Integration and Common Data Model. These tools standardize the data from across the entire toolchain and relate data between systems, creating a high-fidelity toolchain relating work between the various groups.

2. Management and Orchestration. Management of key functional areas of the toolchain include functions from enterprise planning, release orchestration, deployment orchestration, environment management, and deployment orchestration. These management functions standardize the workflow regardless of the underlying tooling.

3. Decision-Making and Analytics. VSM tools provide role-based visualizations and analytics into the entirety of the development and delivery process. It enables a focus on delivering value and reducing waste, tracking lead time, process times, and gaps.



**Figure 8: Value Stream Management Tool**

The following are example engineering practices for VSM:

• End-to-end release processes are made visible and controlled using a VSM tool. • Real time status of releases is made visible using a VSM tool. This enables proactive decision-making around release scope, schedules, and resources to achieve optimal business objectives.

• A real-time dashboard displays the status of all the releases. • Team members and automated tools can update their release activities in a VSM tool. • A VSM tool accepts changes to release schedules and provides visibility into the impact of changes to releases. • Release dependencies are visible using a VSM tool that automatically generates an impact matrix showing releases, application changes, and their scope.

• Compliance audits are supported by a VSM tool that keeps track of activity (who did what, and when) and supports approval workflows.

• Deployment plans, including task assignments and approval workflows, are captured in a VSM.

• Dependencies between deployment steps are captured in a VSM tool.

• A VSM tool tracks deployment task assignment, notifies responsible parties of their tasks, and provides real time status of deployments.

• Execution times of deployment steps are captured in a VSM tool to support analysis for continuous improvement.

• Issues and incidents occurring during deployment are captured in a VSM tool to support analysis for continuous improvement.

• Shared environment conflicts are managed using a VSM tool that provides visibility into environment availability and supports workflows to request/release environments.

• Capacity planning is supported with a VSM tool that captures environment usage and provides utilization reports.

• Environment configurations are captured in a VSM tool.

• Booking requests for environments are handled using a self-service VSM tool.

• Patching and maintenance of environments are supported by a VSM tool that indicates when environments are down.

• Environment change requests are supported by a VSM tool that can pull environment configuration data from existing CMDB, discovery, deploy, or ITSM tools.

• Dependencies between environments are made visible with a VSM tool.

• Compliance audits are supported by a VSM tool that captures environment changes and approvals.

What Is Needed to Engineer a Value-Stream Management Solution for DevOps?

**The following steps are recommended to engineer VSM:**

1. Conduct a VSM assessment. a. Orchestrate leadership interviews. b. Leverage discovery tools. c. Conduct assessment surveys. d. Run value-stream mapping workshops. e. Analyze assessment data. f. Formulating a strategy and solution roadmap. g. Formulate next steps.

2. Select a VSM solution and conduct a POC. a. Survey available tools. Gartner Magic Quadrant and Forester are good sources for tool comparison information. b. Select a tool or craft a solution with existing tools, design a POC with two or three use cases on one or two model applications. c. Deploy the chosen VSM solution with a select number of initial applications on a trial basis. Monitor the trials and make improvements if needed.

3. Deploy the chosen VSM solution(s) to an initial set of production applications. a. Complete implementing the remaining high-priority use cases needed for deployment. b. Implement training. c. Implement success metrics.

4. Operationalize the VSM solution. a. Use the solution for a select set of production value streams for at least three releases. b. Monitor performance of the VSM solution. c. Conduct retrospectives periodically to determine needed improvements. d. Continue to implement more use cases as needed for the VMS solution to gain acceptance with more applications.

**Summary**

In summary, the answer to the question: “How do you engineer Value Stream Maps and Value Stream Management for DevOps?” has been explained in this white paper.

After an organization has defined goals for the value stream, a current state value stream map is created, detailing lead times, process times, non-value-added times and quality ratios for each stage in the value stream. The values are used to determine bottlenecks that can be targeted for reduction, to meet the goals. A future state value stream map is created that meets the goals and solutions are determined that will meet the reduction targets.

A value stream management solution is selected that can provide end-to-end value stream orchestration, aggregation of metrics and analysis in accordance with the engineering blueprint, and engineering practices.

**Learn More**



The book “Engineering DevOps” can be obtained here: [**mybook.to/engineeringdevops**](http://mybook.to/engineeringdevops)

This white paper and other useful papers, glossary checklists, slides, blueprints, templates and engineering practices can be downloaded from <https://engineeringdevops.com/documents>

About the Author:

**Marc Hornbeek** a.k.a. DevOps\_the\_Gray esq.

[linkedin.com/in/marchornbeek](https://www.linkedin.com/in/marchornbeek)

© EngineeringDevops 2019, 2020

All rights reserved

**What Is Engineering DevOps?**